Developing a Matrix Equation Solver for the HAL-15 Hypercomputer® (Research Proposal)

William S. Fithian
New Horizons Governor's School Mentorship
at Langley Research Center, Hampton, Virginia

under the direction of

Dr. Olaf O. Storaasli
Analytical and Computational Methods Branch
Langley Research Center, Hampton, Virginia

For research mentorship conducted at the:

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Sept 2001-May 2002

Abstract

Current state-of-the-art software used by engineers to solve large systems of simultaneous linear equations is limited in its use of parallel processing by interprocessor communication time. This can become prohibitively expensive as the number of processors used increases. A new, revolutionary, massively parallel FPGA computer, the HAL-15, recently acquired by NASA Langley Research Center, will be used to create a new matrix equation solver that will be tested for solution speed against current matrix equation solvers. It is expected that new technology employed by the HAL-15 is very likely to offer advances in matrix solution speed relative to current solvers.

Introduction

Many real-world phenomena can be modeled using systems of simultaneous linear equations. In linear algebra, a system of such equations involving multiple variables can be represented as a matrix equation (Hildebrand, 1965). For instance, the system of equations:

$$3x_1 + 4x_2 - x_3 = -8 \quad (1)$$

$$2x_1 - 2x_2 + 3x_3 = 15 \quad (2)$$

$$6x_1 + 9x_2 + 2x_3 = -6 \quad (3)$$

is represented as the following matrix equation of the form $\mathbf{Ax} = \mathbf{b}$:

$$\begin{bmatrix} 3 & 4 & -1 \\ 2 & -2 & 3 \\ 6 & 9 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -8 \\ 15 \\ -6 \end{bmatrix}$$

where matrix **A** contains the coefficients of the equations. **A** is multiplied by vector **x**, which contains the three unknowns in the equation. This multiplication yields vector **b**, which contains the values on the right-hand sides of the three equations.

For an individual to solve the above system of equations for all three variables would require the use of algebraic methods of adding and subtracting equations from one another to isolate a variable, and then to substitute that variable into another equation with only two variables. The value of another variable could then be found, and both known variables could be substituted into any original equation to ascertain the value of the final variable.

To illustrate the algebraic method, an individual would first multiply equation (2) by 3/2 and subtract it from equation (1) to get equation (2)', then multiply equation (3) by 1/2 and subtract it from equation (1) to obtain equation (3)':

$$3x_1 + 4x_2 - x_3 = -8 \qquad (1)$$

$$0x_1 + 7x_2 - 5.5x_3 = -30.5 \qquad (2)'$$

$$0x_1 - 0.5x_2 - 2x_3 = -5 \qquad (3)'$$

Next, the individual would multiply equation (3)' by 14 and add it to equation (2)' to obtain equation (3)" with only one variable:

$$3x_1 + 4x_2 - x_3 = -8 \qquad (1)$$

$$0x_1 + 7x_2 - 5.5x_3 = -30.5 \qquad (2)'$$

$$0x_1 + 0x_2 - 33.5x_3 = -100.5 \qquad (3)''$$

The matrix now has a triangular form; that is, all coefficients under the diagonal are zero. Triangular matrices are by nature easy to solve (Stewart, 2000). Now it is possible, by dividing both sides of equation (3)" by -33.5, to find that $x_3 = 3$. If we substitute this value for x_3 in equation (2)', we can easily find that $x_2 = -2$. Finally, substituting the values of x_3 and x_2 into equation (1), we find that $x_1 = 1$. Hence, the three values obtained, x_1, x_2, and x_3, are the solutions to the system of equations. They can be represented as the following vector (one-dimensional matrix):

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}$$

A computer program would use this algorithm, called Gaussian elimination, to solve the matrix (though it would not necessarily use the identical process), isolating variables by eliminating terms of the matrix so that all the coefficients under the diagonal equal zero, and the matrix is triangular. Next, the computer program would “backsolve” the matrix as demonstrated above by working backwards, solving the last equation, which has only one variable, first, and then subsequently substituting and solving for values up to the first equation (Storaasli, 1996). This type of algorithm is known as “matrix decomposition” because the computer algorithm actually decomposes the matrix into factors that can be solved more easily. There are many varieties of this approach, and the most popular decomposition algorithms are Cholesky Decomposition, Pivoted LU Decomposition, QR Decomposition, Spectral Decomposition, Schur Decomposition, and Singular Value Decomposition. However, all of these algorithms are essentially variations of the same general method (Stewart, 2000).

Matrix equations occur in many fields of study, due to the fact that many different kinds of problems require the solving of systems of simultaneous algebraic equations. Two examples of such problems include structural analyses to compute the wing deflections for a high-speed aircraft and to compute the failure modes of the Challenger space shuttle solid rocket booster (Storaasli et. al., 1990). Systems of equations that accurately model extremely complex real-world physical phenomena may contain not only three, but thousands or even millions of equations and variables (Storaasli, 1996). Computer programs are the only practical method to compute the solutions of such systems.

The GPS (General Purpose Solver) program developed at NASA Langley is a good example of an efficient matrix equation solver. The GPS solver is used widely by structural engineers in structural analysis. For one structural engineering analysis, GPS calculated the structural deformation of a Ford Thunderbird automobile as a result of frontal impact. The automobile was modeled using 253,574 equations, and 6.3 million coefficients (Storaasli, 1996). Figure 1 shows the car model to illustrate just how complex the application was (Anonymous, unknown date).

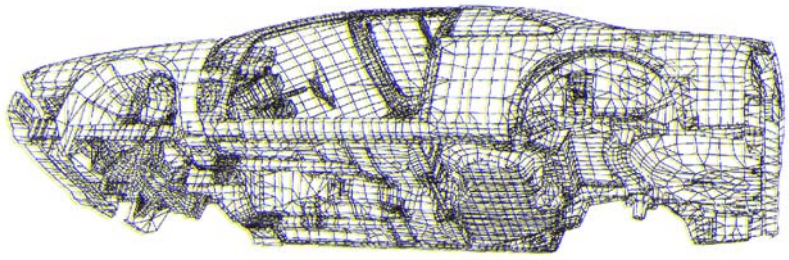

Fig. 1 - Ford Thunderbird Automobile

Current matrix-equation solvers utilize what is called "parallel processing" in their algorithms to improve speed. With parallel algorithms, computers running the software can work on multiple parts of a problem simultaneously by assigning different portions to different processors working together to solve the matrix. Over the years, the fastest equation solvers, including those made by NASA, have been the ones whose algorithms were developed to capitalize on the full capabilities of parallel supercomputers (Storaasli, 1996).

This type of parallel computation, however, has limitations. As the number of processors increases, interprocessor communication time becomes very large and eventually is the dominant time cost. Due to this factor, only a limited number of processors can be used, limiting the number of operations that can be performed simultaneously. The most recent NASA equation solver, the VSS (Vector Sparse Solver), reports its performance on a maximum of eight processors (Storaasli, 1996).

However, NASA has recently acquired a new computer with the potential to perform much speedier operations. This computer, the Star Bridge Systems HAL-15 Hypercomputer, offers hardware that can be programmed. The chips it uses are called field programmable gate arrays (FPGAs), which can be reconfigured each time a user desires to run a new problem to fit the special needs of that particular problem. FPGAs are not brand new, but they are currently the method of choice for creating application-specific integrated circuits (ASICs), machines specially created for a specific application. These machines, which are inherently massively parallel and optimized at a hardware level for their specific tasks (processes wired into the hardware of a system are usually

the fastest processes to complete), are capable of outperforming the best serial microprocessors, both in speed and in space used on the chip (Anonymous, 2001). However, according to the Star Bridge Systems web site, "it usually takes years of education and practice to master the art" of programming them (Anonymous, 2001).

Star Bridge Systems' technological breakthrough lies in the new Viva programming language, which greatly simplifies the task of programming application-specific machines onto FPGAs, bringing it within the reach of the common programmer. This language allows a user to program complicated algorithms at a high level of abstraction in a highly intuitive, graphical user interface (GUI) environment. When the user is finished programming, Viva's compiler then configures the hardware of the computer to perform the task specified by the user, translating the high-level mathematical language of the user program into low-level bit operations which the user need never worry about. Because the user is free from these low-level considerations, he can concentrate on high-level mathematical algorithms and program much more complicated processes much more easily than he could have done before (Anonymous, 2001).

The image on the next page shows the Viva environment in color. The user can simply think about where the data should go and what operations should be performed on it in each step. By simple pointing, clicking, and dragging of the mouse, tubes are created carrying the data into and out of boxes, which are abstracted representations of functions. When the "play" icon is clicked, Viva programs the FPGA chip or chips to create a machine optimized for the user's task; this machine can perform as many operations in

parallel as the FPGA chip has space, potentially hundreds per chip – and computers may contain multiple FPGA chips (Anonymous, 2001).

This new massively parallel potential raises the possibility of an even faster matrix equation solver than any previously written. Consider, for instance, a one thousand by one thousand (or, more generally, an n by n) matrix. A massively parallel computer could achieve the first column of zeros for every row of a matrix simultaneously - this method could be completed nine hundred ninety-nine (n-1) times faster than an algorithm which achieves each row's first zero sequentially. The full triangularization of the matrix, using the parallel algorithm, could be completed five hundred times (n/2) faster than using the sequential algorithm.

Therefore, with this greatly improved degree of parallel capabilities, it seems likely that a matrix equation solver programmed on the HAL-15 could potentially be much faster than the current state-of-the-art solvers. Such a development would be of great practical use. In addition, it is of interest to the experimenter because it involves the use of computer programming to solve complicated mathematical problems.

Methods and Materials

Star Bridge Systems' new Viva programming language will be used to create a matrix-equation solver on a HAL-15 computer acquired by NASA. To begin, several very simple functions which perform basic vector operations were programmed in Viva for the dual purpose of allowing the experimenter to become familiarized with the language and

creating a basic library for use in future, larger programs. These small programs included, among other functions, functions to find a dot product of two four-element vectors, to multiply a four-element vector by a scalar value, to split a four-element vector into its elements, and to collect four numbers into a four-element vector. These programs all had very simple algorithms and were well-suited to learning the basics of the Viva language, which was new to the experimenter.

These and other smaller functions will be combined with Viva basic library functions to create a prototypic matrix-equation solver capable of solving, by Gaussian elimination (the algorithm reviewed in the introduction), a four-by-four matrix given a four-element right-hand-side vector. This prototypic program is nearly complete. The figures on the next three pages show the two major components of this program - one function which triangularizes the matrix and another which backsolves to attain a solution. Flow charts of the top-level program, the matrix triangularization program (nearly finished), and the backsolution program (finished and tested) are included as Figures 2-4. Of note is that the Viva programs which describe these functions are themselves very similar to flow charts.

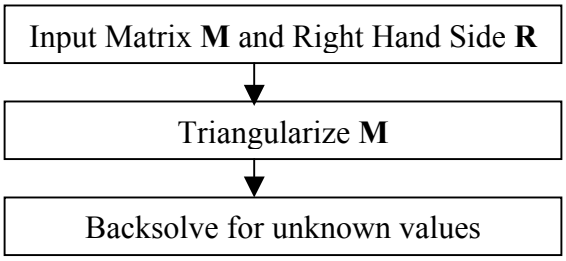

Figure 2: Flow Chart of Top-Level Program

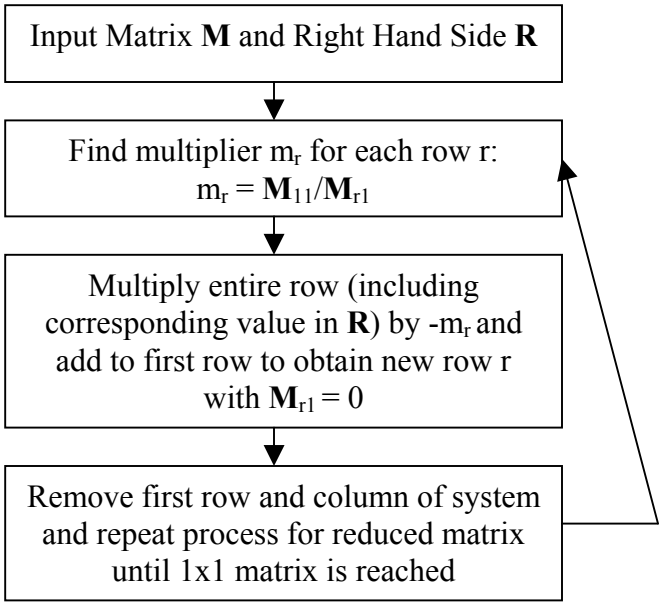

Figure 3: Flow Chart of Triangularize

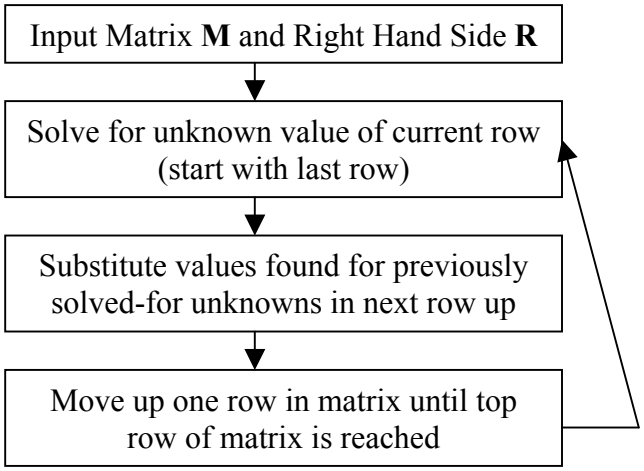

Figure 4: Flow Chart of Backsolve

After the prototype is complete, the program will be generalized to handle an arbitrary-sized matrix, and eventually an arbitrary-sized large matrix (i.e. hundreds or thousands of equations). This process will probably require some creative use of recursion to represent the matrices and the functions which operate on them, but any prohibitively intractable problems are unforeseen at this point. Finally, programs will be written

(unless someone at NASA has already written them by then) to enable the program to read in its matrices from files of the type NASA uses to represent them.

Throughout the designing of the program, optimization of the program with respect to computation and compilation speed will be sought. In particular, old and new methods for eliminating unnecessary multiplications and additions of zeros will be investigated to speed up computation for matrices whose elements are nearly all zeros.

As a final step, the solution time for the new matrix equation solver developed in the Viva language for the HAL-15 will be tested against the current NASA state-of-the-art equation solver to determine whether a significant speed-up has been attained.

Schedule:

December 4, 2001 -	Proposal will be turned in.
December 2001 - February 2002 -	Prototypic program will be generalized for use on large matrices as defined by files in use by NASA.
February - April 2002 -	Program will be tested against current state-of-the-art matrix-equation solvers and modified to achieve maximum computation speed.
May 2002 -	Oral Presentation will be prepared and delivered.

Expected Results

Due to the massively parallel capabilities of the new technology, it is expected that eventually a matrix equation solver using Viva software to program FPGAs will

outperform current state-of-the-art solvers using microprocessors. However, the solver programmed by the experimenter in Viva will probably not take full advantage of all the HAL-15's capabilities due to limitations of time and expertise of the programmer. Still, it is a possibility that the new Viva solver will be the fastest ever, in a best-case scenario. No matter what, however, a great deal will be learned by the experimenter about linear algebra, parallel computing, and computer programming in general.

Conclusions/Relevancy

If the new Viva matrix equation solver is faster than current solvers, an inevitable conclusion will be that the technological breakthroughs made by Star Bridge Systems have indeed made possible faster matrix equation solvers than have ever been created before. However, a failure to create a faster matrix equation solver will not be interpreted as a failure of the HAL-15, nor will it discredit the potential of the new technology involved. If the new Viva solver is the fastest equation solver ever written, it will be very relevant to engineers who will be able to solve matrices faster than ever before and it will also imply that the HAL-15 has great potential in many mathematical and engineering applications.

Acknowledgments

I would like to thank my mentor, Dr. Olaf Storaasli, for generously sharing his time, expertise, and enthusiasm with me. I would also like to thank Richard Loosemore and Samuel Brown at Star Bridge Systems for their help with the Viva language, and I thank NASA and Star Bridge Systems for the privilege of working with this exciting new technology. Finally, I would like to thank my mother for helping me with editing.

Literature Cited

Anonymous. Unknown date. Equation solution performance records for automobile model. <http://transit.larc.nasa.gov/csb-www/AUTO.html>

Anonymous. 2001. Star Bridge Systems Web Site. <http://www.starbridgesystems.com>

Hildebrand, F.B. 1965. Methods of Applied Mathematics. Dover Publications, Inc., New York. 362 p.

Stewart, G.W. 2000. The decompositional approach to matrix computation. Computing in Science and Engineering January/February 2000: 50-59.

Storaasli, O.O., Nguyen, D.T. and Agarwal, T.K. 1990. A parallel-vector algorithm for rapid structural analysis on high-performance computers. Hampton, VA: NASA Langley Research Center. NASA Technical Memorandum 102614.

Storaasli, O.O. 1996. Performance of NASA equation solvers on computational mechanics applications. <http://techreports.larc.nasa.gov/ltrs/papers/NASA-aiaa-96-1505/olaf.fm5.html>